J. Symbolic Computation (1985) **1**, 245–260

Automated Inferencing

W. BIBEL

Duke University, Durham, North Carolina, USA
and
Technische Universität, München, Germany

(*Received* 29 *May* 1985)

This article provides a survey of aspects and methods of the automation of processes involved in performing inferences. It addresses classical deductive reasoning as well as non-monotonic and inductive reasoning. Several important reasoning methods are informally described and illustrated. The emphasis is on a clarification of some of the main principles rather than on an exhaustive listing of particular techniques.

Introduction

Intelligent systems rely on knowledge and inference as their two basic building components. It is the inference component, in particular, that provides a qualitative distinction between a classical system (such as a computer's operating system) and one that features a behaviour in some limited domain of application which, compared with that of humans, might be called intelligent. Namely, a lot of knowledge is coded within an operating system; due to the lack of the capacity for inferencing it cannot make intelligent use of it, but rather performs what it is programmed to do in a functionally fixed, unintelligent way.

Strictly speaking, the capacity for inferencing may equally well be regarded as knowledge, that is knowledge *about* how one can infer new knowledge from previous knowledge; we might thus better speak of *meta*-knowledge in this case. As such it is distinct from all the knowledge on the *object*-level, consisting of data along with their functional and relational interrelations.

A confusingly rich variety of methods and techniques for inferencing is known today. It ranges from mathematical theorem proving to the speculative conclusions of a stockbroker in one dimension, and from the human forms to sophisticated machine versions in another. So while this survey attempts to provide some informal insight into the nature of automated inferencing, it by no means covers all possible aspects let alone the different approaches to cope with each of them.

We shall take the position here that all kinds of inferencing can, and as a strategy for further research should, uniformly be viewed as controlled deduction. By deduction we mean deductive inference in classical logic, thereby thinking mainly of first-order logic along with higher-order features (see sections 1.5 and 2.2). To some extent the control is expected as being built into the deductive engine, to some other extent explicitly provided as knowledge on the meta-level, depending whether it handles a speed-up facility for

0747–7171/85/030245+16 $03.00/0

special situations (like algebraic rewriting) or has to cope with the fact that knowledge is imperfectly stated, most of the time, in real systems. This view splits the present paper naturally into two parts.

In the first part, we consider deduction under the classical paradigm in Automated Theorem Proving (ATP) of perfect information available for the deductive process. In particular, resolution (1.3) and the connection method (1.4) are described—these approach the problem from orthogonal points of view (see section 1.4). Natural deduction (1.4), extensions to higher-order and other logics (1.5), representational issues (1.6), and build-in control (1.6) are briefly addressed.

In the second part, we discuss the measures that have to be taken, if the knowledge provided to the system does not exactly match the world that is meant to be modelled. This gives rise to various types of non-monotonic reasoning (2.1). It also raises the problems of quoting in the stated knowledge other knowledge; these kinds of problems are discussed under the usual keywords of knowledge and belief (2.2). Inductive and abductive reasoning is required for the same reason of lacking complete information (2.3). All these issues are regarded as accessible by deductive means along with extra information that controls the deductive steps in an appropriate way. To be fair, we clearly point out the existence of approaches taken from different perspectives as far as space allows.

Given this huge program, the reader might consider material beyond the present article, such as Loveland (1984) reviewing ATP and Bibel (1984*a*, 1985*b*) covering more material on the topics related with the second part. Also, for reasons of space the list of references does not include some of the work mentioned in the text. It may easily be looked up in the references of Bibel (1982) a book the reader might find useful for more comprehensive studies anyway.

It is no longer necessary to argue that the automation of the processes involved in performing inferences, automated inferencing for short, lies at the heart of symbolic computation. In a sense any other symbolic (in fact even numerical) computation may be viewed from this perspective. Some people argue that this universality is just what dims the prospects of success for this particular area. We believe that on the contrary the uniform and to some extent universal view taken in automated inferencing is one of its most attractive features. The experimental results achieved in recent years indicate that these features have not to be traded off with effectiveness and to some extent even efficiency. But we realise that much remains to be done.

1. Controlled Deduction

In this first part of the paper we survey logical formalisms along with their deductive mechanisms. Here we adopt the paradigm of classical deduction that assumes the relevant knowledge being stated in a formalised and complete way. Various deductive reasoning forms are uniformly viewed as controlled deduction with varying control features.

1.1 FIRST-ORDER LOGIC

For more than 2000 years logic has been the science for the development and study of formalisms for the representation of knowledge and inference. Suppose our knowledge consists of the two sentences: "everybody has a father" (K1 for short) and "a grandfather is a father's father" (K2 for short). Then clearly the knowledge in "everybody has a

grandfather" (K for short) is logically implied by K1 and K2, in the natural sense of this term.

A number of issues are raised with this simple example, some of which are mentioned now. One issue is the relationship between sentences like K1, K2, and K on the one hand and the knowledge (or meaning), that is carried along with them, on the other. Its study has been initiated by Frege, Russel and others, and is now carried on in the field of natural language semantics and in model theory. We will briefly address this issue in the second part of the paper.

Another issue concerns the formalisation of the natural language sentences so that ambiguities disappear and conceptual differences become apparent. Again Frege is one of the pioneers in this endeavour, creating what we call first-order logic. The previous three sentences may be formalised in its language in the following way.

K1 $\forall u \; \exists f \;\; Ffu$

K2 $\forall xyz \; (Fzy \wedge Fyx \rightarrow Gzx)$

K3 $\forall c \; \exists v \;\; Gvc.$

If we read $\forall u$ as "for all u", $\exists f$ as "there exists an f", $\wedge$ as "*and*", $\rightarrow$ as "implies", Ffu as "f is father of u", Gzx as "z is grandfather of x" and the rest in an analogous way, then the sentences obtained by reading the formulas this way would be naturally considered to carry the same meaning as the original ones. This illustrates the requirement that the semantic content should not be affected by the formalisation.

Note that our syntax tries to avoid the burden of too many parentheses and commas. Only in the case of potential confusion would we therefore write $F(f, u)$ instead of Ffu.

The next issue is concerned with formalising the notion of logical implication. This is, in the first place, a relation between a set of sentences (like the set consisting of K1 and K2) and another set, usually a single sentence (like K). There are two different ways of defining this relation, the *semantic* and the *syntactic*.

Usually, the semantic way is regarded as the more natural one, in which we define this relation (denoted by $\models$) in terms of the meaning of the sentences. So in our example we would think of any set of people of the sort specified by K1 and K2, and see that the property specified by K necessarily holds for such a set as well. Formally, we write

$$\text{K1, K2} \models \text{K}$$

in this case.

One may question what it really means to say that this is a *semantic* way of definition. Namely, this way requires a formal definition of the relation $\models$ which cannot be given but in terms of writing down a bunch of symbols, hence again something of a purely syntactic nature. Under this view one would exclusively rely on a syntactic way of definition as a relation (then denoted by $\vdash$) on the set of strings of symbols. In this case we write

$$\text{K1, K2} \vdash \text{K}.$$

Herbrand was among those who have taken this purely syntactic point of view (favoured also by the author). Usually, both ways of definition are acknowledged and by way of a completeness and soundness theorem it is assured that the two relations $\vdash$ and $\models$ coincide.

Once the relation $\vdash$ has been defined, which from now on is taken for granted, the question arises as to how one could determine for any two sets of formulas whether one logically implies the other; moreover, is there perhaps even a mechanical way so that a computer could carry out this task. Here is where automatic deduction enters the stage.

1.2 REPRESENTATIONAL VARIANTS

Most deductive mechanisms are defined for formulas or expressions in some representational form different from the one illustrated with K1, K2, and K. Let us consider some of these forms before we turn the attention to the mechanisms themselves.

First of all we mention that (for closed formulas) K1, K2 $\vdash$ K holds if and only if $\vdash$ K1 $\wedge$ K2 $\rightarrow$ K holds. Note that the latter is of a special form in so far as it relates the *empty* set of strings with the string K1 $\wedge$ K2 $\rightarrow$ K (FG for short). The transformation to this special case, as illustrated with our example, may be done this way in general, which restricts the deductive problem uniformly to the case of a single given formula.

Many (but not all) mechanisms are actually testing for unsatisfiability rather than validity of the given formula; this means that they process the negation of the formula, such as $\neg$(K1 $\wedge$ K2 $\rightarrow$ K), or, equivalent, K1 $\wedge$ K2 $\wedge$ $\neg$K in our example. All resolution methods are defined that way (although a trivial change in their definition could avoid the negation).

Most (but not all) mechanisms assume no quantifiers to be present which may be achieved by *skolemisation* and by convention in the use of variables. Skolemisation in our example, for instance, amounts to considering the existentially bound variable f as a function the value of which clearly depends on the u under consideration; so K1 is replaced by $\forall u\ F(fu, u)$ in this way. After skolemisation all remaining quantifiers are deleted by convention.

The deductive methods finally vary in the preferred representation of the resulting propositional formula. The following is a list of (equivalent) possibilities illustrated for our example.

As a set of Horn clauses (for PROLOG):

$$F(fu, u) \leftarrow$$
$$Gzx \leftarrow Fzy, Fyx$$
$$\leftarrow Gvc.$$

As a set of clauses (for resolution):

$$\{F(fu, u)\}, \{Gzx, \neg Fzy, \neg Fyx\}, \{\neg Gvc\}.$$

As a matrix with three rows (for Andrews' matings):

$$F(fu, u)$$
$$Gzx \quad \neg Fzy \quad \neg Fyx$$
$$\neg Gvc.$$

As a matrix with three columns but without negating FG (for the connection method):

$$\neg F(fu, u) \quad \neg Gzx$$
$$Fzy \quad Gvc.$$
$$Fyx$$

We highlight the main points of the differences between these representations. The first is the most restricted one since only a small though practically important class of formulas can be represented in this way as a set of *Horn clauses* or, in other words, in *positive implication form.*

An equivalent *clause form* can always be found for a first-order formula although the usual way of transformation has a serious drawback with respect to efficiency in the subsequent proof process since it expands the given formula in a way so that its length

may grow exponentially. Very recently it has been shown how this drawback can be avoided with a different form of transformation (Minc, personal communication; Eder, 1984; Poole, 1984; Plaisted, 1985).

The matrix form is the least restricted among these since in general the matrix structure may be nested to an arbitrary level. Thus one avoids the drawback mentioned for the clause form. As a matter of fact the method suggested in these papers for avoiding length explosion simply amounts to a more intuitive way of representing the nested matrices by introducing auxiliary names for them. The difference between the last two (matrix) representations clearly is a trivial one and may be regarded as a matter of taste. By far less trivial is the avoidance of Skolem functions in which case essentially the original formula FG is handled by the deductive process. The same is true for natural deduction-type proof techniques, which take the original formula as their input.

1.3 RESOLUTION AND PROLOG

The previous two sections have prepared the ground on which deductive mechanisms operate. As we pointed out their basic goal is to determine whether for a given formula F (or its equivalent expression in any of the various representations) $\vdash$ F holds (i.e. whether F *is a theorem*). By nature of first-order logic such a mechanism is bound to never terminate in certain cases. Only if $\vdash$ F does hold is the finite termination guaranteed. The running systems have nevertheless demonstrated their fundamental importance in practice with a number of striking successes (e.g. Wos & Winker, 1984).

There is a confusingly large amount of deductive techniques—ranging in the hundreds. At some level of abstraction, however, essentially three different (but still closely related) methods might be distinguished: the *resolution method*, the *connection method*, and *natural deduction*. These will be briefly discussed in the present and subsequent section.

Resolution clearly is the most popular among these three. It is illustrated with our previous example for which a resolution proof is shown in Fig. 1. Basically, it works as follows. Input is in clause form. Any clause, say $\{\neg Gvc\}$ is selected. Therein any literal, thus here necessarily $\neg Gvc$, is selected. Then, a second clause is selected which contains a literal with the same predicate symbol, which is G in our case, but differs with respect to negation, hence Gzx in our case. If possible a new clause, the *resolvent* of these selected (*parent*) clauses is generated by the resolution operation and added to the previous set of clauses. In some detail this means the following.

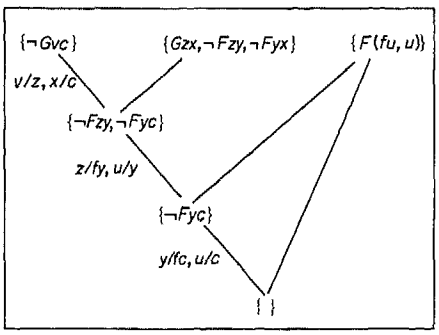


Fig. 1. A resolution proof for FG.

The respective terms of the two selected literals are *unified* by way of an appropriate substitution of the occurring variables; here we substitute z for v and c for x. The substitution determined this way is carried out in all literals of the two clauses. The two *complementary* literals $\neg Gzc$ and Gzc are removed and the remaining ones collected in the new clause.

This process is repeated until the empty clause is generated which signals the successful proof. In a justification of this whole process we have to prove that $\{\}_\Delta$ trivially is unsatisfiable, and that unsatisfiability is not affected by adding resolvents.

Note that clauses are never deleted and so backtracking is never required (i.e. the relation defined by "adding resolvents" is a *confluent* one). From that point of view the various selections in the process are not crucial for an eventual success as long as the control is such that each possible resolution step eventually gets a chance to be executed. They are essential, however, from the point of view of efficiency since an awkward choice may result in an unnecessarily long proof. So the issue actually amounts to a search for a hopefully short proof in the space of all possible resolution proofs, in general a complex space indeed.

In fact the *general* results are more than depressing. Even if we restrict ourselves to formulas in propositional logic, then it has recently been shown that the length of the shortest proofs may be exponential in the worst case (Haken, 1984), let alone the search in order to find them. Specialised results in contrast to this are much more encouraging.

For instance, the unification to be performed in each resolution step may be done in linear time. This is particularly encouraging since experience with running systems shows that unification is still the most time consuming part of the whole process. Another encouragement is due to the programming language PROLOG which essentially is processed by a specialised resolution prover, as we may see again with our example.

Here the process starts with the unique goal clause which has no literal to the left of the arrow, i.e. with $\leftarrow Gvc$ in our case. Its literals are put on the *goal list*. The left-most one is selected, Gvc in our case, and attempted to be unified with one of the *heads* of the *program* clauses, i.e. with their left-most literals, one after the other from top down. In our case the match fails with the head $F(fu, u)$ of the first clause but succeeds by way of unification with that, viz Gzx, of the second one. The selected goal Gvc is now replaced in the goal list by the literals Fzy and Fyc in the *body* of that clause after performing the substitution. This process is repeated until the goal list becomes empty.

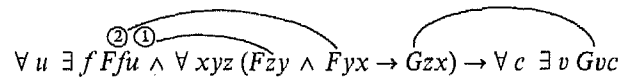
A comparison with the resolution proof above shows that both are different representations of the same process, at least in this particular example. Note, however, that the process now is strictly determined by the sequence of clauses and their literals since no selections are made here. This requires that in PROLOG back-tracking has to be used. Also a proof that exists may actually never be found. Moreover, PROLOG has relaxed the requirements on unification (by skipping the so-called *occur-check*), and thus may even result in wrong "proofs". But in most cases it works and even does so in a remarkably efficient way.

A feeling why PROLOG is so efficient may be obtained by once more considering the restricted case of propositional logic where we easily see that a proof may be found in at most time $m \cdot n$ where m is the number of clauses and n the number of different propositional variables. This is so because at least one clause necessarily must contain a head only (i.e. an empty body) to be provable. By at most m unit resolution steps all occurrences of the variable from this head may be eliminated from the set of clauses. Doing this for all n variables guarantees the empty clause.

The language of Horn clauses is remarkably rich in its expressiveness. Moreover, non-Horn clauses to some extent can be transformed into Horn ones by *renaming* negated predicates with new (non-negated) predicates and by more complicated processes like the one described in (Caferra *et al.*, 1984). Nevertheless the restriction often does not allow one to state a problem in a natural way. Hence PROLOG can only be regarded as a step in the right direction.

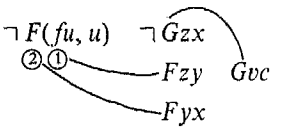
1.4 THE CONNECTION METHOD AND NATURAL DEDUCTION

The connection method (not to be confused with connection graph resolution) attempts to determine $\vdash F$ for any given formula F by an analysis of the structure of F without any change in it. For instance, the formula FG from section 1 is recognised as a theorem in this approach by the identification of three pairs of literals, depicted as connections, along with a substitution as follows.



$$x\backslash c,\ y\backslash f.2,\ z\backslash f.1,\ u.1\backslash f.2,\ u.2\backslash c,\ v\backslash f.1.$$

As we see the proof can be carried out without a single change in the given formula. Yet the flavour of the process may be more easily understood if we explain it in terms of the matrix representation from section 1.2 where it reads as follows.



$$x\backslash c,\ y\backslash f(c),\ z\backslash f(f(c)),\ u.1\backslash f(c),\ u.2\backslash c,\ v\backslash f(f(c)).$$

First of all, the index labels 1 and 2 code the fact that the connected literal $\neg F(fu, u)$ has to be regarded in two different instances, one for each of these two connections, viz. $\neg F(f(u.1), u.1)$ and $\neg F(f(u.2), u.2)$. Then we see that after performing the substitution all connected literals are complementary as it has to be. A second requirement for qualifying as a proof is that each *path* through such a matrix (say, from left to right) contains two connected literals. A path is obtained by selecting a single literal from each clause (i.e. column) of the matrix whereby each clause has to be considered as often as the indices require. For instance, $\{\neg F(fu.1, u.1), \neg F(fu.2, u.2), Fyx, Gvc\}$ is one of the three paths through the present matrix, and it clearly satisfies the requirement since its second and third literal are connected in the matrix.

Elaborate procedures have been developed which test for these two requirements. The one concerning the substitutions is very much the same as in resolution although special aspects have to be taken into account (Eder, 1985*a*). The other one may be tested by merely moving pointers through the matrix in some controlled way. Both work very similarly if the connection proof is carried out in the original formula rather than in the matrix representation.

In Bibel (1985*a*, *b*) the class of *linear* connection proofs has been introduced that allow any instance of a literal to occur in at most one connection. Linear connection proofs have similar characteristics in terms of efficiency to Horn clause proofs. As a matter of fact, in the case of Horn formulas the proofs may essentially be carried out by the linear

connection method. However, since linear connection proofs are applicable to formulas beyond Horn ones, they characterise a wider class of formulas provable in a relatively efficient way, which might be the most promising class beyond Horn for any practical purposes like programming, plan generation, etc.

In a brief comparison of the connection method with resolution, the first one might note is the representational advantage of the connection method in terms of computer memory needed for a proof; this becomes apparent if one compares the previous connection proof with the resolution proof in Fig. 1. Note in this comparison that each resolution step corresponds to a connection in this simple example. Structure sharing copes to some extent with this problem for resolution.

As we mentioned in section 1.3 the space of all possible resolution proofs has to be taken into account as the search space. There are a number of different derivations of the empty clause by resolution already for our simple example, while our matrix above shows that the depicted connections are the only possible ones in this configuration. At least this demonstrates that the search space is more transparent in the case of the connection method. This transparency of the connection method in all respects might in fact be regarded as its most attractive feature.

The strength of resolution lies in its capability to encode whole proofs in the form of a lemma; $\{\neg Fyc\}$, in the proof of Fig. 1, for instance, encodes the part of the proof that produces this clause. Such a lemma may be used more than once in the remaining proof. The way we explained the connection method above this method does not enjoy this property, as can be seen using the theorem

$$Pc \wedge \forall x\,(Px \to Pfx) \to Pf^n c$$

for some n (cf. Eder, 1985*b*). This feature may, however, be included by taking advantage of sets of connections as a whole more than once in a single proof. With this extra feature the connection method has an advantage in comparison with resolution and other approaches. Note that this is the author's opinion, but it is based on extensive studies of comparisons of various known proof techniques.

In some sense the resolution and connection methods approach the deduction problem from viewpoints that may be regarded as orthogonal to each other, since resolution focuses the attention *on* the clauses while the connection method emphasises the relations *between* the clauses (cf. Davydov, 1974). Despite that fact both methods might eventually converge in their performance.

Just as resolution stands for a whole family of techniques, the same is true for the connection method; in particular, Andrews' matings and Maslov's inverse method are very closely related. This family actually evolved from a study of the calculi of natural deduction of Gentzen along with all their derivatives (like the tableau method). It is therefore no surprise that there is a close connection between a connection proof and a proof in any of these calculi. More precisely, there are straightforward algorithms that provide the translation from one into the other (see section IV.8 in Bibel, 1982).

This feature is particularly important for an interactive use of such a deductive tool, where the user is given (possibly partial) proofs in a natural deduction style only while the machine takes full use of the coded proof mechanism like resolution or the connection method. The recurring attempts to focus the attention on natural deduction for the purpose of improved performance of the machine are therefore misled and ignore the historical development of proof techniques.

1.5 EXTENDED LOGICS

First-order logic encodes features of reasoning of so fundamental a nature that it seems unlikely that there might ever be a logic that would not include it in some way or another. So in considering other logics we are actually talking about variants, restrictions, or extensions. We have already mentioned restrictions imposed either on the structure of the formulas or on the deductive mechanism, where the latter kind is probably the more reasonable one (if one keeps in mind that systems are interactive).

Among the extensions by far the most important is the one to *higher-order* logic. Consider the notion of any two objects (mathematical or real ones) being equal. The most natural meaning of this notion is that there is no property (among those in question) by which they may be distinguished. Such a natural definition cannot be expressed in first-order logic since expressing the existence of a property would require a quantification over a predicate that is not allowed in it. This can only be done in second-order logic. This is but one among a variety of examples (taken from mathematics, natural language understanding, knowledge representation, etc.) that demonstrate the naturalness of higher-(than-first-) order logic. Note that we emphasise the naturalness since from a theoretical point of view there is actually no need to go beyond first-order logic.

In spite of its attractiveness from various viewpoints, higher-order logic is not at all popular, and this is a pity. True, in its full generality this logic is by far too rich, complex, and computationally infeasible. For instance, the issue of unification that can be decided in linear time as we mentioned in section 1.3, is not even any more decidable in second-order logic. But this only calls for carving out an adequate part of it rather than dismissing it altogether. All attempts into such a direction seem not to have resulted in a convincing solution (monadic logic, etc.). We feel that the characterisation of such a part should be attempted from the point of view of the deductive structure (like the linearity restriction in the previous section) rather than from that of the formula structure considered in previous attempts.

Nevertheless, such future work can build upon a rich source of existing results. For instance, the connection method generalises to higher-order logic in a straightforward way (see section V.6 in Bibel, 1982), while for resolution such a generalisation has not been found so far. This fact is another indication for the higher degree of transparency of the connection method in comparison with resolution mentioned in the last section.

One of the virtues of the classical logics considered so far lies in their neat separation of the representational aspects (i.e. the language) from the deductive and control aspects. A whole bunch of logics have been developed which sacrifice this virtue. One way of interfering the representational aspects with control features is by way of introducing extra operators with a semantics that is dynamic by nature. The modal operators $\Box$ and $\Diamond$ belong to this category which is one of the reasons why the author does not sympathise too strongly with these kinds of approaches in view of the present stage of development. This is not meant to discourage the interested reader to study such approaches (e.g. Farinas, 1982; Fitting, 1983) in more detail.

1.6. REPRESENTATION AND CONTROL

Apparently, the deductive methods discussed so far are neutral with respect to the particular application for which they might be used, be it in an expert system for mineral exploitation or in a proof system specialised for linear algebra. They constitute the sophisticated result of a process of abstraction that lasted over many centuries. Although

the way back from the level of abstraction to the particular application is short, it nevertheless has to be taken. It consists in providing special factual and control knowledge to the deductive systems based on any of the previous methods, along with a little thought on representation.

Consider, for instance, a knowledge base that includes the following facts:

Bc	for	*the cardinal is a bird*
$Bx \to Ax$	for	*birds are animals*
$Ay \to Iyo$	for	*animals inhale oxygen.*

We would like to have a system realise a deducted fact like *cardinals inhale oxygen* without each time explicitly running through the chain of deductive steps that are shown in the following (linear connection) proof.

$$Bc \wedge (Bx \to Ax) \wedge (Ay \to Iyo) \to Ico.$$

In Artificial Intelligence semantic nets have been invented to produce such a behaviour. They do in fact provide a quick remedy for this particular purpose but, not surprisingly, lack all the other powerful features of the methods presented before. On the other hand these methods do provide these extra features at no cost with representational techniques used in ATP probably before the introduction of semantic nets, namely structure sharing and precomputed deduction. We illustrate this by representing the above knowledge base along with the two connections in the form of a dag (directed acyclic graph) (Fig. 2). The precomputed inferred knowledge (generated by the two connections) is represented by the two nodes labelled with an asterisk. Assume we want to know some property enjoyed by cardinals. Start at c, and move upward; any of the three arcs leads you to a node representing one of the properties: Bc, Ac, Ico.

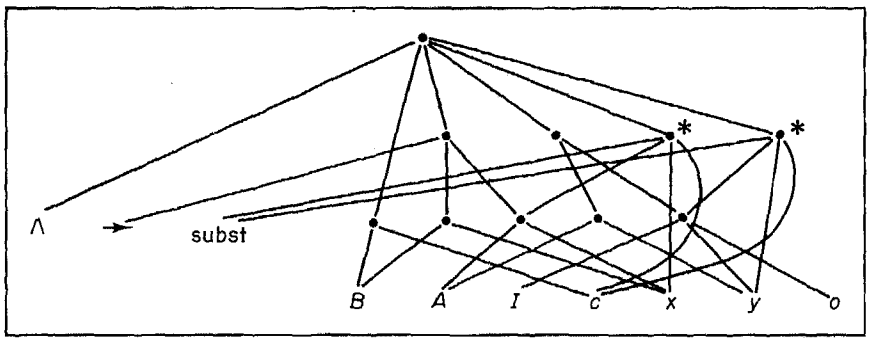


Fig. 2. Directed acrylic graph (see text for explanation).

This demonstrates two important issues. First, the way logical formulas are represented affects the deductive performance, of course. Dags are appropriate data structures for that purpose, and allow for what today is called object oriented representation. Second, precomputation of chains of deductive steps obviously improves the response time of a given system at the price of a relatively small increase of storage space.

There are other ways of speed up in the response time of deductive systems. For instance, assume the system specialises in group theory. The uniform approach taken by resolution or the connection method would mean that the system is provided with the axioms of group theory and equality. Experience shows that without extra guidance such

a system performs too many unproductive steps, which a mathematician specialising in group theory would never even take into consideration.

This observation has led to what is known as the critical-pair/completion procedure (Knuth & Bendix, 1967/1970; Buchberger, 1965/1970; see also Buchberger, 1985). This procedure generates from a given set of axioms a list of rewrite rules that prove a given formula by rewriting it in a fixed way determined by this list of rules. Since we have a preference for our uniform approach rather than for a scattered one that calls for the application of a special method in each particular case, we would like to embed the advantage of this procedure within our deductive methods. This is straightforward along the following two lines.

We take the view of the uniform approach. But we use the result of this procedure as a standard *control* for the sequence of deductive steps (cf. section V.4 in Bibel, 1982). Along the second line we encode standard sequences of deductive steps, that involve frequently used axions (like the equality axioms), in form of macro-steps ("macro-" connections in the case of the connection method; cf. section V.3 in Bibel, 1982). Some of these macro-steps may be viewed as unification in a generalised form. For instance, the term $3+5$ would immediately unify with the term 8 without any regress to the axioms that define numbers and addition. But note the main point being that all this is achieved as an extra feature, but otherwise without any change of the uniform deductive system.

There are further issues that fall into the topic of the present section. The treatment of *sorts* is one more of them. It has a particularly simple solution since it is simply a *preference* in the control for the connections leading into the sort statements that settles it. For instance, Schubert's well-known steamroller problem, meant to be a challenge in this context, collapses to a trivial deductive problem.

More interesting is the question of proofs by induction. In particular for programming, induction is of extreme importance. Theorem proving by induction, as a special subfield of ATP, is planned to be treated in a separate survey in a future issue of this journal. Here we just mention the most successful system supporting proofs by induction, which was developed by Boyer and Moore.

2. Inference in an Imperfect World of Knowledge

If we think of systems that are supposed to reason under real circumstances, then the paradigm adopted for the first part of this paper must be seen as an ideal that will rarely be achieved. In this second part we therefore briefly address some of the issues that arise in much more complex situations. Basically, all complications may be viewed under the common aspect that the knowledge available for performing the desired inferences is imperfect in various ways. Essential knowledge might not be stated, pieces of knowledge might contradict each other, or the knowledge might be defined in a vague and fuzzy way only. Deficiencies of that sort apparently have nothing to do with the logic of reasoning itself. Our goal therefore must be to accommodate the precise methods from the first part of this paper to these extra complications rather than to start over again in the search for a new logic.

2.1. NON-MONOTONIC REASONING

The formal systems of logic addressed in the first part of the paper enjoy the following *monotonicity property*. If a piece K of knowledge follows from some knowledge $K2$, then K also follows from $K1$ enriched by additional knowledge $K2$. In the proof-theoretic

notation from 1.1:

$$K1 \vdash K \quad \textit{implies} \quad K1, K2 \vdash K.$$

Common sense reasoning often is *non-monotonic* in this sense. For instance, we all take it for granted that *birds can fly*. Suppose we also learn that *penguins are birds*, so that our knowledge base *K1* now comprise these two facts (formalised in some appropriate way). Obviously, we may now infer that *penguins can fly* as an extra piece K of knowledge. At some other occasion, suppose, we realise that penguins are in fact an exception to the rule, which is to say *they cannot fly* (*K2*). If we add this new insight, K obviously can no longer be inferred, or, if we treated *birds can fly* in the sense of an all-quantification, the knowledge base *K1, K2* becomes contradictory.

This paradigmatic example gives the flavour of the kinds of problems that arise in non-monotonic reasoning. McCarthy (1984) distinguishes the following seven different uses which we might have in mind when talking about non-monotonic logic.

1. As a communication convention which requires only the exception (*penguins cannot fly*) to be stated.
2. As a knowledge base convention that certain predicates are to be understood as having their minimal extension (in the sense of model theory).
3. As a rule of conjecture to cope with expressions of the sort: *most birds can fly.*
4. As a representation of a policy like for: *the meeting will be on Wednesday unless another decision is explicitly made.*
5. As a very streamlined expression of probabilistic information when numerical probabilities are unobtainable like for: *she is a young and pretty woman.*
6. For auto-epistemic reasoning like: *if I had an older brother, I'd know it.*
7. In common sense physics and common sense psychology.

This shows us that we are dealing here with a wide-spread phenomenon. Basically, it arises when the facts stated explicitly do not match exactly a complete description of the world under consideration, but, by way of using the facts appropriately ("with common sense"), nevertheless allows us to arrive at the right conclusions. "Using the facts appropriately" calls for some control mechanism that is imposed on top of the general deductive mechanism. For instance, this extra control is supposed to allow the use of the fact *birds can fly* in all bird-like circumstances, except if penguins are mentioned. Such a control (expressing common sense behaviour) by nature is a meta-level feature, i.e. knowledge that talks about the factual knowledge. There are various ways of integrating such meta-level knowledge into a knowledge base.

One way consists in stating such control knowledge explicitly in a separate meta-level language and in providing means so that the reasoning on the meta-level is in harmony with that on the object level. Weyrauch (1980) and Bowen & Kowalski (1982) have taken this approach. The deductive mechanism itself is usual first-order reasoning as described in section 1.

Another way to state such meta-level knowledge is provided by higher-order logic. Such an approach has been taken by McCarthy (1980, 1984). He has proposed a scheme that may be instantiated for any given set of facts. This instantiation added to the knowledge base as an additional axiom restricts the deductive system so that entities satisfy a given predicate only if they have to on the basis of the set of facts. In this sense circumscription is a kind of minimisation. Note that it is achieved without any change of the deductive means except for a proof system in second- rather than first-order logic being required.

The circumscription approach recently has attracted considerable attention so that, for instance, a substantial fraction of the papers in a recent workshop on non-monotonic reasoning was exclusively concerned with it. It carries some inconvenience, though, since it requires a special way of stating knowledge such as the one about birds. We cannot simply say, *all birds fly*, but have (for instance) to provide for an extra predicate ABNORMAL that distinguishes between the normal and the abnormal cases under various aspects. At first sight, at least, this appears awkward. Also such a requirement is not needed in the meta-level approach.

According to what we said for modal logic in section 1.5 it is clear that this logic is also a candidate for handling the kind of issues raised in the present section. But clearly the reservations expressed there apply here as well. Again we emphasise, however, that our judgement has to be understood as a high-level strategy towards achieving certain scientific goals rather than as an individual judgement of such excellent work like Reiter (1980) and Moore (1983), that we would like to mention in this context.

The issue of non-monotonic reasoning occurs already in more restricted settings in PROLOG, where *negation-by-failure* (Clark, 1978) carries its flavour, and in data-bases, where the *closed-world assumption* (Reiter, 1978) causes non-monotonic effects.

Among the uses of non-monotonic reasoning listed further above we mentioned the expression of probabilistic information. Predicates like YOUNG apparently have a fuzzy meaning. Zadeh (1979) has proposed to model this phenomenon by way of a logic the truth values of which vary in the closed interval between 0 and 1. This would clearly be an attractive approach, which has even caught the attention of pure logicians (Takeuti & Titani, 1984), if in practice we only had the right numbers determining the truth in each particular instance.

2.2. KNOWLEDGE AND BELIEF

Since 1900 logicians have made several attempts in finding the right way to represent and reason about knowledge and belief. The most natural way of formalising a sentence like *I know all that you said is not true* seems to be

$$KNOW(I, \text{`}\forall\, x\, [SAID(you,x) \rightarrow \neg\, TRUE(x)]\text{'}).$$

In fact, this is essentially what Frege had in mind when it all started. Unfortunately, Russel showed that such an approach leads to inconsistencies in the underlying deductive system. As a remedy he proposed type theory or higher-order logic, the importance of which has been pointed out in section 1.5. But we also mentioned its immense complexity. Moreover, there are natural sentences like *John has no religious beliefs* that find no natural way of representation in type theory either (Perlis, 1985).

Therefore, efforts have continued to discover a consistent way of formalizing what natural language seems to accomplish at ease. These efforts have been strengthened by the vital interest of Artificial Intelligence in these questions, resulting in a variety of new proposals recently. One particularly intriguing proposal has been made by Perlis (1985), that seems to cope with many (if not all) previous problems and is appealing in its simplicity.

It allows formulas like the one above without entering the second-order level simply with a quotation, as natural as one always wanted. At the same time he avoids the troubles experienced before with this attempt with a simple change in Tarski's "No truth-definition theorem". This theorem says that $TRUE(\text{`}A\text{'}) \leftrightarrow A$ (what one would naturally expect for the predicate *TRUE*) is inconsistent, which causes all the troubles. Now Perlis

proposes a weaker form $TRUE(`A') \leftrightarrow (A)^*$, where the * operator replaces each connective occurrence of the form $\neg\, TRUE(`\ldots') \text{ IN } A$ by $TRUE(`\neg(\ldots)')$, and proves the consistency of the resulting formalism. In this he builds upon work by Gilmore, Kripke and McCarthy.

Note that a reflection principle of some sort is needed to relate the content of the quoted sentences with the unquoted parts in a chain of reasoning. Perlis' proposal might be just what had been missing for nearly a century. It opens the perspective to reason about knowledge and belief in a purely first-order setting. The same applies to control and other meta-level concepts that talk about the expressions on the object level. Since a final judgement might be premature, let us also mention recent work of Bibel (1984*a*), that is similar in spirit but by far not as profound, as well as that of Haas (1983) and Levesque (1984), where the latter takes a different approach in the spirit of Hintikka's possible-worlds semantics.

Any of these proposals are able to cope with the complications in the reasoning process that arise in situations as the one in our concluding example. Suppose, *Par knows Mike's phone number which is incidentally the same as Mary's*. Obviously, an inference system must not conclude that Pat knows Mary's number unless there is additional evidence that Pat *knows* about the coincidence. This might give a feel for the subtleness of the issues involved here, that do actually occur in a variety of applications (e.g. think of distributed knowledge bases).

2.3. INDUCTIVE AND ABDUCTIVE INFERENCE

Inductive inference attempts to derive a complete and correct description of a phenomenon from specific observations of that phenomenon or of parts of it. Inductive inference lies at the heart of the evolution of any science. For instance, take the most formalised discipline, mathematics, as an example. Suppose a child has learned to add numbers and to distinguish even from odd ones; then it observes in a number of examples that two odd ones always add up to an even one; by inductive inference it may quickly be convinced that this holds in general.

By inductive inference only hypotheses can be generated, which raises the problem of their validation. In mathematics this can be achieved to some extent by a proof or disproof, if such can be found. In natural sciences such hypotheses cannot, in principle, be completely validated (except in special cases) because they may have an unlimited number of consequences, a fact which gives sciences their evolutionary character.

Inductive inference plays a key role in learning and knowledge acquisition. The same is true for the general problem of program construction. That is why this particular kind of inference has recently begun to attract rapidly growing attention. As a matter of fact, programming may be regarded as the paradigmatic test-bed of inductive inference. Since Biermann (1985) has provided a tutorial from this very aspect recently in this journal we may confine ourselves here with a very few remarks, but emphasise once more the importance of this particular branch of inference (see also Bibel, 1985*b*).

Often, inductive inference is contrasted with deductive inference, thus suggesting that different techniques might be needed for both tasks. While it is true that a deductive system as such does not enjoy inductive capabilities without any extra provision, it is also true, however, that deduction plays the key role also in inductive inference. Namely, inductive inference is based on a number of principles, that, formalised and added to the knowledge base of a deductive system, enable it to inductively infer hypotheses from facts and these principles in a purely deductive way.

An instance of such a principle is the inductive rule $X \rightarrow X \wedge Y$. Namely, suppose the fact $P \wedge Q \rightarrow K$ is given; an often used inductive inference generalises this fact to $P \rightarrow K$. With the rule above this generalisation may be established in a purely deductive way, since $X \rightarrow X \wedge Y$ and $P \wedge Q \rightarrow K$ (deductively) implies $P \rightarrow K$ (X, Y being propositional variables that will be instantiated to P, Q by unification).

Note that the inductive rule $X \rightarrow X \wedge Y$ is the inverse of the *logically valid* formula $X \wedge Y \rightarrow X$. This observation is true for any inductive rule. As another example consider the logically valid formula $\forall x\ F(x) \rightarrow F(t)$ for any term t. Its inverse $F(t) \rightarrow \forall x\ F(x)$ is not logically valid, and thus may be used as another useful inductive rule. This observation may be exploited to minimise the number of different inductive rules, since we know that very few logically valid formulas may be taken as the generative kernel of all other valid formulas.

We mention the Model Inference System (Shapiro, 1982) as one of the most remarkable systems capable for inductive inference. It takes a number of facts as input. It keeps generalising as long as some facts are not *covered* by the inferred hypotheses. At the same time it tests for counterexamples with a well-known deductive technique for that matter. Once a counterexample is detected this indicates that the generalisation has gone too far. By tracing the cause, which produces the counterexample, and eliminating it, the hypotheses are weakened in their generality in a tuned way, as is the subsequently resumed generalisation process. This incremental procedure comes to a halt if all facts are covered and counterexamples are no longer produced. The system has been applied especially to program debugging and synthesis of PROLOG programs. But its mechanism is a general one, applicable to many other areas.

With respect to inductive inference realized for the purpose of learning systems, Michalski *et al.* (1983) provide a rich source of relevant information. A survey exclusively confined to theory and methods of inductive inference is given in Angluin & Smith (1984).

Abductive inference is a way of reasoning with an inductive flavour that is of particular importance in diagnostic systems that for a given set of manifestations are supposed to explain why they are occurring by postulating the presence of one or more causative disorders.

In comparison with *modus ponens*

given fact A and rule $A \rightarrow B$, infer B,

a deductive rule par excellence, abductive inference may be characterised as

given fact B and association $A \rightarrow B$, infer plausible A.

So the two forms of reasoning seem to differ. But actually, abductive reasoning is backchained deductive reasoning, that is deductive reasoning controlled in a certain fixed way. It starts with fact B and, in view of modus ponens, matches the conclusion of some of the rules $A \rightarrow B$ and thus, by resolution or the connection method, arrives at A as the potential cause in a backward way. The reason why we nevertheless mention this way of reasoning here (rather than in section 1) lies in the fact that in most applications, facts and associations are imperfectly reflecting so that plausibility considerations play a major role in this process. Reggia & Nau (1984) describe one way of formalizing such considerations.

I greatly appreciate a number of discussions with D. Loveland on the topics presented in this paper, as well as his comments on an earlier version resulting in various improvements. The typescript is by A. Davis.

References

Abgluin, D., Smith, C. H. (1984). Inductive inference: theory and methods. *Comput. Surv.* **15,** 237–269.

Bibel, W. (1982). *Automated Theorem Proving.* Vieweg: Braunschweig.

Bibel, W. (1983). Matings in matrices. *Commun. ACM* **26,** 844–852.

Bibel, W. (1984*a*). First-order reasoning about knowledge and belief. In : (Plander I, ed.) *Proceedings of the International Conference on Artificial Intelligence and Robotic Control Systems, Smolenice, CSSR, June 1984.* Amsterdam: North-Holland.

Bibel, W. (1984*b*). Inferenzmethoden. In: (Habel, C., ed.) *Künstliche Intelligenz.* Berlin: Springer.

Bibel, W. (1985*a*). A deductive solution for plan generation (submitted to *New Generation Journal*).

Bibel, W. (1985*b*). Predicative programming revisited. In: (Bibel, W. & Jantke, K., eds) *Proc. MMSSSS'85.* Berlin: Akademie Verlag (in press).

Biermann, A. W. (1985). Automatic programming: A tutorial on formal methodologies. *J. Symbolic Computation* **1,** 119–142.

Bledsoe, W. W. & Loveland, D. W., eds. (1984). Automated theorem proving: After 25 years. *Contemporary Mathematics, vol. 29.* Providence: American Mathematical Society.

Bowen, K. A. & Kowalski, R. A. (1982). Amalgamating language and metalanguage in logic programming. In: (Clark, K. L., *et al.*, eds) *Logic Programming*, pp. 153–172. London: Academic Press.

Buchberger, B. (1965). Ein Algorithmis zum Auffinden der Basiselemente des Restklassenringes nach einum nulldimensionalen Polynomideal. Dissertation (Ph.D. thesis), Universität Innsbruck, see also *Aequationes Mathematicae* **4,** 374–383. (1970).

Buchberger, B. (1985). History and basic features of the critical-pair/completion approach, *Proc. of 1st Internat. Conf. on Rewriting Techniques and Applications, Dijon, May 1985,* Springer LNCS (in press).

Caferra, R., Eder, E., Fronhöfer, B., Bibel, W. (1984). Extension of PROLOG through matrix reduction. In: (O'Shea, T., ed.) *Proc. ECAI-84.* Amsterdam: North-Holland.

Clark, K. (1978). Negation as failure. In: (Gallaire, H., *et al.*, eds) *Logic and Data Bases.* London: Plenum Press.

Davydov, G. V. (1973). Synthesis of the resolution method with the inverse method. *J. Soviet Math.* **1,** 12–18.

Eder, E. (1984). An implementation of a theorem prover based on the connection method. In: (Bibel, W., Petkoff, B., eds) *Proc. AIMSA'84.* Amsterdam: North-Holland.

Eder, E. (1985*a*). Properties of substitutions and unifications. *J. Symbolic Computation* **1,** 31–46.

Eder, E. (1985*b*). Reduction of redundancy of proofs in the Gentzen-Schütte calculus (in preparation).

Farinas del Cerro, L. (1982). A simple deduction method for modal logic. *Inf. Proc. Lett.* **14,** 49–51.

Fitting, M. (1983). *Proof Methods for Modal and Intuitionistic Logics.* Dordrecht: Reidel.

Haas, A. (1983). The syntactic theory of belief and knowledge. Report No. 5368, BBN, Cambridge.

Haken, A. (1984). University of Illinois, Urbana.

Knuth, D. E. & Bendix, P. B. (1967). Simple word problems in universal algebras. In: (Leed, J., ed.) *Proc. of the Conf. on Computational Problems in Abstract Algebra,* Oxford, 1967. Pergamon Press (1970), 263–297.

Kowalski, R. (1979). *Logic for Problem Solving.* New York: North-Holland.

Levesque, H. (1984). A logic of implicit and explicit belief. *Proc. AAAI-84.*, pp. 198–202.

Loveland, D. W. (1978). *Automated Theorem Proving.* Amsterdam: North-Holland.

Loveland, D. W. (1984). Automated theorem proving: A quarter-century review. *Contemporary Mathematics, vol. 29,* pp. 1–45. Providence: American Mathematical Society.

McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artif. Intell.* **13,** 27–39.

McCarthy, J. (1984). Applications of circumscription to formalize common-sense knowledge. *Non-monotonic Reasoning Workshop.* AAAI, pp. 295–324.

Michalski, R. S., Caronell, J. G., Mitchell, T. M. (1983). *Machine Learning.* Palo Alto: Tioga. (Second volume in preparation.)

Moore, R. C. (1983). Semantical considerations on non-monotonic logic. In: (Bundy, A., ed.) IJCAI-83, pp. 272–279. Los Altos: Kaufmann.

Perlis, D. (1985). Languages with self-reference I: Foundations. *Artif. Intell.* **25,** 3.

Plaisted, D. (1985). A structure preserving clause form translation (submitted).

Poole, D. L. (1984). Making "clausal" theorem provers "non-clausal". *Proc. CSCSI/SCEIO Conf., London, Ontario,* pp. 124–125.

Reggia, J. A., Nau, D. S. (1984). An abductive non-monotonic logic. *Non-monotonic Reasoning Workshop.* AAAI, pp. 385–395.

Reiter, R. (1978). On closed world data bases. In (Gallaire, H., *et al.*, eds) *Logic and Data Bases.* New York: Plenum Press.

Reiter, R. (1980). A logic for default reasoning. *Artif. Intell.* **13,** 18–132.

Shapiro, E. Y. (1982). *Algorithmic Program Debugging.* Cambridge, Mass.: The MIT Press.

Takeuti, G., Titani, S. (1984). Intuitionistic fuzzy logic and intuitionistic fuzzy set theory. *J. Symbolic Logic* **49,** 851–866.

Wos, L. & Winker, S. (1984). Open questions solved with the assistance of AURA. *Contemporary Mathematics, vol. 29,* pp. 73–88. Providence: American Mathematical Society.

Weyrauch, R. W. (1980). Prolegomena to a theory of mechanized formal reasoning. *Artif. Intell.* **13,** 133–197.

Zadeh, L. (1979). A theory of approximate reasoning. *Machine Intelligence, vol. 9,* pp. 149–194. Amsterdam: Elsevier.